ISSN:2231-0711
www.ijcset.net

Text Extraction from Images

Paraag Agrawal[#1], Rohit Varma[*2]

[#]*Information Technology, University of Pune, India*
[1]paraagagrawal@hotmail.com
[*]*Information Technology, University of Pune, India*
[2]catchrohitvarma@gmail.com

***Abstract*— Automatic image annotation, structuring of images, content-based information indexing and retrieval are based on the textual data present in those images. Text extraction from images is an extremely difficult and challenging job due to the variations in the text such as text scripts, style, font, size, color, alignment and orientation; and due to extrinsic factors such as low image contrast (textual) and complex background. However, this is realizable with the integration of the proposed algorithms for each phase of text extraction from images using java libraries and classes. Initially, the pre-processing phase involves gray scaling of the image, removal of noise such as superimposed lines, discontinuities and dots present in the image. Thereafter, the segmentation phase involves the localization of the text in the image and segmentation of each character from the entire word. Lastly, using the neural network pattern matching technique, recognition of the processed and segmented characters is done. Experimental results for a set of static images confirm that the proposed method is effective and robust.**

***Keywords*— Image Pre-processing, Binarization, Localization, Character Segmentation, Neural Networks, Character Recognition.**

I. INTRODUCTION

Nowadays, information libraries that originally contained pure text are becoming increasingly enriched by multimedia components such as images, videos and audio clips. They all need an automatic means to efficiently index and retrieve multimedia components. If the text occurrences in images could be detected, segmented, and recognized automatically, they would be a valuable source of high-level semantics. For instance, in the Informedia Project at Carnegie Mellon University, text occurrences in images and videos are one important source of information to provide full-content search and discovery of their terabyte digital library of newscasts and documentaries [1]. Therefore, content-based image annotation, structuring and indexing of images is of great importance and interest in today's world.

Text appearing in images can be classified into: Artificial text (also referred to as caption text or superimposed text) and scene text (also referred to as graphics text). Artificial text is artificially overlaid on the image at a later stage (e.g. news headlines appearing on television, etc.), whereas, scene text exists naturally in the image (e.g. the name on the jersey of a player during a cricket match, etc.) [2]. Scene text is more difficult to extract due to skewed or varying alignment of the text, illumination, complex background and distortion. This paper focuses on artificial text and its extraction from still images.

Existing OCR engines can only deal with binary text images (characters against clean background), and it cannot handle characters embedded in shaded, textured or complex background [3]. This is not always the case, as there exists many disturbances (noise) in the input text images. These disturbances have a high influence on the accuracy of the text extraction system.

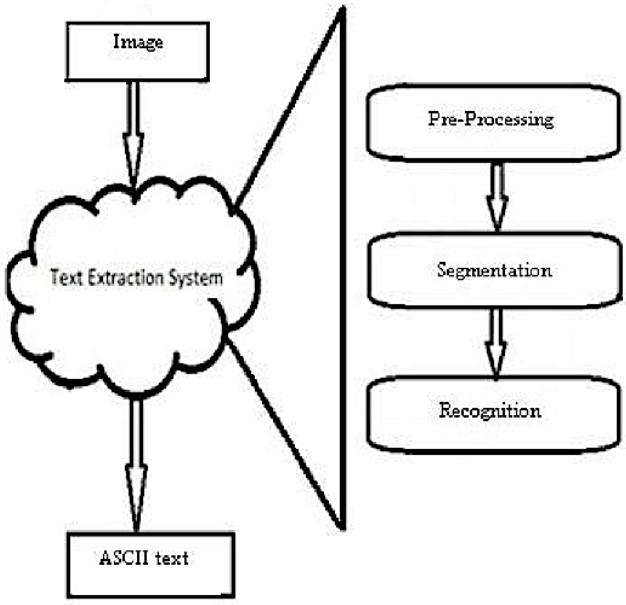

Fig. 1 Our proposed model

The process of extracting text from images consists of various stages as seen in Fig. 1. Each stage consists of steps that are explained with the help of select algorithms in Section II and are finally demonstrated by presenting experimental results for a set of static images in Section III.

II. METHODOLOGIES

The input image to our proposed system has a complex background with text in it. The first stage is image pre-processing, which serves to remove the noise from the input image and generates a clear binary image. Text segmentation is the next stage, where we differentiate each character from the entire word by circumscribing them into boxes and saving them each separately. The final stage is text recognition, where the segmented characters are compared to the stored character matrices and as a result, the closest match for each character is displayed.

A. Image Pre-processing

The main purpose for pre-processing the image is to produce an image containing the text to be recognized without any other disturbing elements or noise. This step contributes significantly to boost the performance of text recognition. To remove the disturbances and noise (i.e. everything else except the text) from the image, gray scaling is done firstly, followed by line and discontinuity removal, which is lastly followed by dot removal.

1) Gray scaling: The main purpose for gray scaling is to produce a binary image (containing of black or white pixels only), thus, making it easier to distinguish text from the background. First, all the pixels in the image are converted to shades of gray. To do this, the RGB (R: Red, G: Green, B: Blue) color components of each pixel in the image are extracted using bitwise shift operators. Each of these values of the R, G and B components vary from 0-255 [4]. Then, these values are added in a proportion of Red: 30%, Green: 59% and Blue: 11% [5] to get the gray scaled equivalent of that particular pixel. This method is applied to each pixel in the image to convert the entire image into gray scale.

The technique used for text localization and segmentation (discussed further) was developed for binary images only, and thus, cannot be applied to colored images directly. The main problem is that the colored images have rich color content and complex colored background, having high resolution with a lot of noise, which makes it difficult to localize the text and segment the characters from those images.

Thus, the gray-scaled image is then converted to a binary image using simple binarization techniques such as gray scale thresholding [6]. Binarization is one among the many conventional methods used for text extraction from images. These methods are based on the assumption that text pixels have a different color than the background pixels. Thus, a threshold color value is used to separate the text from the background. To be specific, this is done by comparing each pixel value to a threshold value (that lies between black and white) and setting that pixel value to black or white as its consequence. This yields the gray-scaled binary image that is the input for the next step of pre-processing. This step also provides a means for the storage of pixel values of the entire image in an array for further processing.

It shall be noted that the noise and disturbances contained in an image might share the same color as the text, and thus, makes it difficult to detect the exact region containing the text in the image. Thus, line removal; discontinuity removal and dot removal need to be performed for successful segmentation of text.

2) Line removal: Noise, in the form of horizontal fluctuation (a horizontal line throughout the image) or vertical fluctuation (a vertical line throughout the image) might have been introduced in the image. Thus, it is necessary to remove these horizontal or vertical lines from the image. To do so, the image is scanned progressively to detect rows and columns having black pixels in the entire width and height respectively. The pixel color values of the entire row or column detected are changed from black to white, thus, removing the fluctuation from the input image. It is important to note that this step does not affect the images having no horizontal or vertical lines.

3) Discontinuity removal: Line removal leads to the generation of discontinuities (conversion of black pixel to white) in the text at locations that were earlier intersected by lines that were removed during the line removal step. This makes the recognition process difficult and needs to be rectified by filling in the discontinuities (converting of white pixels back to black) generated in the text. To achieve this, 8-connected pixel connectivity is used. 8-connected pixels are neighbors to every pixel that touches one of their edges or corners. These pixels are connected horizontally, vertically, and diagonally. Each of the pixels in the rows and columns of lines eliminated during the line removal step are scanned and their neighbouring pixels are taken into consideration. If the diagonally, vertically or horizontally opposite pairs of the neighbouring pixels are black, then the pixel under consideration is also set as black. This is because the pair of black neighbouring pixel indicates that the pixel under consideration was converted to white at the line removal stage, due to the line passing through that pixel. This is performed iteratively until all the pixels, lying on the coordinates of the lines removed, are processed successively. Thus, the broken characters in the text due to the line removal step are successfully filled in again.

4) Dot removal: This is the final step of pre-processing, wherein the remaining disturbances (noise) like unwanted black pixels (dots) are eliminated. An assumption here is that each of the clusters of black pixels forming the noise is significantly smaller than the clusters of any of the characters of the text. Here, the entire image is scanned first by column, then by row, and the first black pixel that is found is taken into consideration. Using 8-connected pixel connectivity, a count is maintained, viz. incremented for every black pixel that is found connected to the pixel under consideration. This process is iterated until all the black pixels in that particular cluster have been added to the count. The count should not be incremented more than once for the same pixel, viz. possible using simple stack operations such as push and pop. This count of number of black pixels contained in a particular cluster is then compared to a default threshold value. Lesser the count than the threshold value indicates that the cluster under consideration is some unwanted noise. In this case, all these connected black pixels are converted to white, thus eliminating the noise from the image. Please note that the characters in the text are also considered as clusters of black pixels, but are unaffected since the count of the connected black pixels in their case is much higher than the default threshold value.

B. Text Localization and Segmentation

After we finish pre-processing the input image, all that remains is the text against the plain background. To separate each character from the entire word in the image, localization of individual characters is done, thus

segmenting the text and generating distinct windows for each and every character in the image.

Text localization involves circumscribing the characters of the text one after the other. To do so, a three-line horizontal group scan is performed from left to right on the pre-processed image until a black pixel is encountered [7]. All the connected black pixels in that particular cluster (first character of the text for the first iteration) are converted to green and the minimum and maximum X and Y co-ordinate values are stored in an array [1]. Again, using 8-connected pixel connectivity and simple stack operations, such as push and pop, all the connected black pixels in that cluster are discovered and the extreme X and Y coordinate values are extracted. Here, the change of color is done in order to make sure that the character localized earlier is not re-encountered and that the next character is directly taken into consideration in search for the next black pixel. This process is iterated for all the characters until the scan reaches the end of the image without encountering a single black pixel, i.e. when all the black clusters of characters have been converted to green and localized.

C. Character Recognition

Now that we have discrete segmented character arrays localized from the pre-processed image, we arrive at the last stage of text extraction from images, i.e. recognition of characters. This involves the thinning and scaling of the segmented characters, followed by matching of those characters with the stored neural network character matrices (generated using MATLAB software) and displaying the closest match found.

1) Thinning and Scaling: The principal reason for thinning is the skeletonization of binary images with applications in the areas of shape representation and matching. This involves the usage of 8-connected pixel connectivity for the successive deletion of boundary pixels until a single 8-connected one pixel wide component is obtained, which approximates the medial lines of the original image [8].

The thinned characters are scaled to an aspect ratio of fixed dimensions of pixels preserving the shape of the characters. The main purpose for scaling is to improve recognition of text of smaller font sizes as well as to save time for recognition of text of font sizes larger than that fixed dimensions of the pixels [1].

2) Comparison with stored neural network matrices: The processed character matrices are compared with the stored neural network character matrices and the closest match is displayed as the recognized text. Here, the technique involves the transformation of the processed image's pixel array into a binary weighted matrix of fixed dimensions. As a result, uniformity is established in the dimensions of the input and stored patterns of characters [9]. Each candidate character stored in the neural network possesses a corresponding weight matrix. The weights of the most frequent (black) pixels are higher and usually positive and those of the uncommon ones (white pixels) are lower and often negative. Therefore, importance is assigned to the pixels based on their frequency of occurrence in the pattern. In other words, highly probable pixels are assigned higher priority while the less frequent ones are penalized [9]. The transformed input weight matrix is matched to the weight matrix of each of the candidate characters stored in the neural network. To do this, the sum total of the product of input pattern and the corresponding elements of the stored weight matrix, is divided by the sum total of all the positive elements of the stored weight matrix of the current candidate character. This yields the probability of the match of the input pattern with the current candidate character [9]. This process is iterated for all the stored candidate characters and the one yielding the highest probability using the neural network matrices is declared as the recognized character.

III. EXPERIMENTAL RESULTS

The proposed methodology in this paper is tested by building software in Java using java libraries and classes. The system accepts an input image with complex background and applies the set methods presented in this paper to recognize the text from the image. Fig. 2 through Fig. 9 demonstrate the successful step-by-step extraction of the text "school", from the input image (Fig. 2).

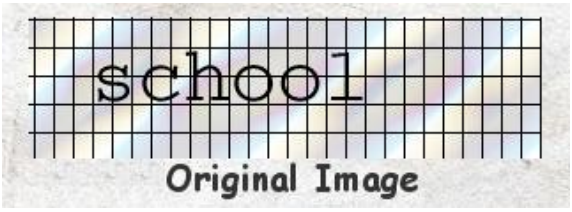

Fig. 2 Input Image

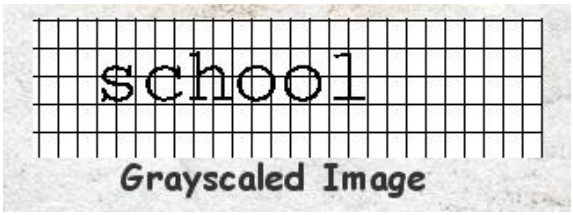

Fig. 3 Gray Scaled Image

Fig. 4 Line Removal

Fig. 5 Discontinuity Removal

Fig. 6 Dot removal

Fig. 7 Segmented Characters

Fig. 8 Thinned and scaled characters

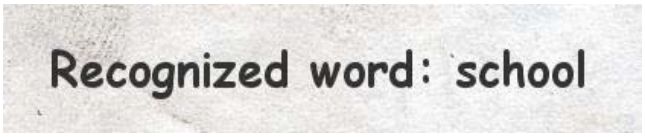

Fig. 9 Recognized word

Here's providing another example to suffice that the proposed methodology is robust and works well with a set of static input images. Fig. 10 through Fig. 17 demonstrate the successful extraction of the text "face", from the input image (Fig. 10).

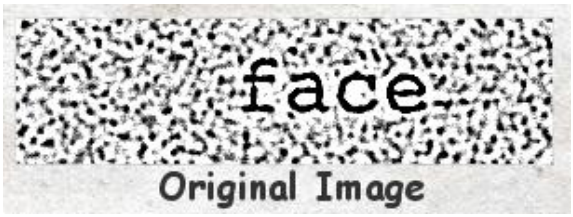

Fig. 10: Input Image

Fig. 11 Gray Scaled Image

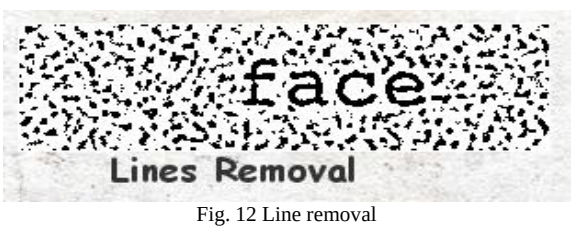

Fig. 12 Line removal

Fig. 13 Discontinuity removal

Fig. 14 Dot removal

Fig. 15 Segmented characters

Fig. 16 Thinned and scaled characters

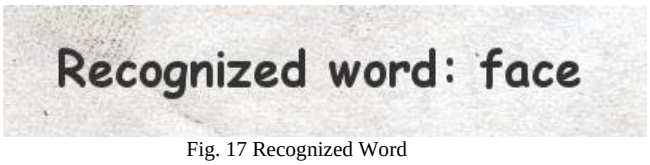

Fig. 17 Recognized Word

IV. Conclusions

This paper introduces a text extraction system and discusses a methodology with promising directions for future research. This methodology has been successfully tested for a particular font in ASCII text (English language). This can be further extended to other fonts and scripts.

Few applications of this system include: (1) Automatic number plate recognition (ANPR), (2) Content based image retrieval: Searching for the required keywords in the images and retrieving the images accordingly, (3) Document data compression: From document image to ASCII text, etc.

There might be a few cases in which certain characters may not get recognized correctly, i.e. a character may be recognized as some other character. This might be due to the discrepancy in the pixel intensity values of the processed input matrix and the various stored character matrices. It can be inferred from the experiments performed on the software that most of the text gets recognized successfully and that the proposed methodology is robust and efficient.

ACKNOWLEDGMENT

The authors would like to thank Prof. P. A. Bailke and Mr. Shekhar P. Khakurdikar for their constructive comments and insightful suggestions that improved the quality of this manuscript. Success is the manifestation of diligence, perseverance, inspiration, motivation and innovation. Every work begins with a systematic approach leading to successful completion.

REFERENCES

[1] R. Lienhart and A. Wernicke, *Localizing and Segmenting Text in Images and Videos*, Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 4, April 2002.

[2] C. Wolf and J. M. Jolion, *Extraction and Recognition of Artificial Text in Multimedia Documents*, Lyon research center for Images and Intelligent Information Systems.S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[3] S. Jianyong, L. Xiling and Z. Jun, *IEEE: An Edge-based Approach for Video Text Extraction*, International Conference on Computer Technology and Development, 2009

[4] J. Gllavata, Extracting Textual Information from Images and Videos for Automatic Content-Based Annotation and Retrieval, PhD thesis, Fachbereich Mathematik und Informatik der Philipps-Universität Marburg, 2007.

[5] C. Bunks, *Grokking the Gimp*, New Riders Publishing Thousand Oaks, CA, USA, 2000.

[6] E. K. Wong and M. Chen, *IEEE: A Robust Algorithm for Text Extraction in Color Video*, International Conference on Multimedia & Expo. (ICME), 2000.

[7] K. Subramanian, P. Natarajan, M. Decerbo and D. Castañòn, *IEEE: Character-Stroke Detection for Text-Localization and* Extraction, Ninth International Conference on Document Analysis and Recognition (ICDAR), 2007.

[8] R. Mukundan, Binary Vision Algorithms in Java, International Conference on Image and Vision Computing (IVCNZ), New Zealand, 1999.

[9] S. Araokar, *Visual Character Recognition using Artificial Neural Networks*, Neural and Evolutionary Computing, Cornell University Library, 2005.